Sept 13th

minimal progress loss → BACKUP WORK + COMMIT TO AVOID LOSS.

Collab Process

- each collaborator creates a new branch
- provide process for merging their branch into master
  - ↳ PULL REQUEST (merge repo + my remote version)

- FORK REPO
- MAKE ALL CHANGES TO YOUR COPY
- Request via PULL REQUEST

☆ KEEP YOUR COPY UP-TO-DATE

☆ ALWAYS CREATE A NEW BRANCH AND DON'T COMMIT TO MASTER (will be easier to rebase)

rebase the branch → PULL REQUEST.

WORKFLOW

FORK → CLONE (cd new dir) → git remote add upstream <link> → git fetch upstream → git rebase upstream/master → git push origin master (now even w/ upstream) → git checkout -b <branch-name> (creates /switches to new branch) → code . (open editor ↵ vscode) → [CHANGE] → git diff → git status → git add <files> → git commit -m <msgs> → git log → git push origin <branch-name> → go to github → open pull request → git fetch upstream → git rebase origin master → git push origin/master

GIT COMMANDS

[-h flag for help]

git log (q to exit)

git status

git diff (difference b/w orig + new changes)

☆ AVOID git add . (adds all files in directory)

git add <filename>

git commit -m "<descriptive msg>"

git push origin <branch-name>

git remote -v (find all remote connections)

↳ git fetch origin (requires git rebase origin/master)

↳ git push

git rebase -i <commit id> → grab from git log.

(interactive → open vim → wq to exit after removing last commit)

(requires git push origin master -f) → force required.

→ for removing commits.

Clean Code Notes.

- structure
- method
  - top down - good for type checked lang.
  - bottom up - testable at each point
  - get it done, get it right, get it fast
- general
  - Keep code as simple/organized as possible.
  - Don't commit to specificity → generalize
  - many small funcs > one large func.
  - read/anticipate your code's output.